BELLCOMM, INC.

SUBJECT: Transmittal of Bell Telephone Laboratories Memorandum Entitled, "Project Apollo - Communications Processor - Analysis of Processing (P_1) Function," by A. E. Peterson, dated November 22, 1965. Case 320

DATE: January 4, 1966

FROM: J. J. Hibbert

TO: Distribution

The attached memorandum by Mr. A. E. Peterson of Bell Telephone Laboratories is an analysis of part of the processing function performed by the Communication Processor at MCC-H. This is another portion of the analysis of the Communications Processor to determine its ability to handle a peak load of data during Gemini missions. It was found that the margins against data loss were not markedly reduced even with a combined live and simulation load.

2021-JJH-rc

J. J. Hibbert

Attachment

Copy to
Messrs. T. A. Keegan/MA-2 ← COPY TO
W. E. Miller/MOG
J. E. Quinn/MOG

H. E. Clements/MSC
R. L. Holt/MSC
C. C. Kraft/MSC
D. T. Myles/MSC (4)
J. Satterfield/MSC

L. Stelter/GSFC

J. P. Downs
L. A. Ferrara
D. R. Hagner
P. L. Havenstein
J. A. Hornbeck
J. E. Johnson
H. Kraus
C. R. Moster

I. M. Ross
G. H. Speake
T. H. Thompson
R. L. Wagner

Central File
Department 1023
Library

BELL TELEPHONE LABORATORIES
INCORPORATED

SUBJECT: Project Apollo - Communications Processor - Analysis of Processing (P_1) Function - Case 20061

DATE: November 22, 1965

FROM: A. E. Peterson

MF5-4332-54

SUMMARY

The Univac-490 Communications Processor at the Mission Control Center at Houston, Texas, has been analyzed for its capability to process a peak Gemini data load. The analysis technique has been to develop a logical model which represents the way in which the Communications Processor performs its highest priority processing task. This is followed by the application of a Gemini data load to the model.

The highest priority Processing task specified by the model is the recognition that a certain complete segment or frame of data is available in the Communications Processor. An integral part of this task is the temporary storage of several identifying parameters associated with the specific frame of data. This task has been classified as the first level of Processing or P_1. Additional analysis work is required to develop the lower levels of Processing, which have been classified as P_2, P_3, and P_4. These levels operate upon the data frame, through the appropriate worker program, and arrange for proper data output.

[redacted]

The results of the P_1 analysis have been expressed in the form of the margins against data loss. When the live Gemini mission load alone was applied to the algorithm, or model there was no degradation in the margin against data loss. However, when the simulation load was added to the live load, there was some reduction in the margin. This margin reduction had no significant effect on systems performance.

BELL TELEPHONE LABORATORIES
INCORPORATED

SUBJECT: Project Apollo - Communications Processor - Analysis of Processing (P_1) Function - Case 20061

DATE: November 22, 1965

FROM: A. E. Peterson

MF5-4332-54

MEMORANDUM FOR FILE

1. Introduction

The Communications Processor (CP) at the Mission Control Center - Houston (MCC-H) handles digital data traffic flowing during manned spaceflight missions. The CP consists of a Univac-490 computer surrounded by various peripheral equipment. The subject of this memorandum is the analysis of the capabilities and limitations of the CP, in the active support of Gemini missions.

In the initial phase of this analysis work, it was decided to separate the overall function of the CP into three categories: Transfers, Storage, and Processing. The Transfer function performs the task of transferring data characters between the Communication Line Terminals (CLT's) and designated buffer areas in core storage in the Univac-490 computer. The Storage function essentially "takes over" where the Transfer function ends. Specifically, the Storage function is primarily concerned with the time usage of the aforementioned buffer areas in core storage.

Analyses have been conducted of both the Transfer and Storage functions. The results of these analyses have been documented in memoranda.[1, 2]

The remaining function to be analyzed is the Processing function. The Processing function essentially takes over where the Storage function terminates. To illustrate: assume that a buffer has been filled with a complete message segment or frame of data. When the CP has accumulated a complete frame, it must first recognize this fact, then decide what to do with the frame or message, and ultimately initiate action for its appropriate disposition. This then is considered as the overall Processing function.

It has been convenient to break the overall function of Processing into four levels. The first level is designated P_1, and performs the task of recognizing that a certain complete segment of data is available to the CP. The second level (P_2) performs a "real-time" analysis of the segment or message. The purpose of this analysis routine is to determine which of the several communications worker programs should get this message for further processing. The third level (P_3) is the actual handling or processing by the chosen worker program. An example of the activities in this category would be the conversion from the data format found on the communication circuits to the data format required by the Real Time Computer Complex (RTCC). There is a fourth level (P_4) of Processing which performs various communications housekeeping tasks such as controlling the traffic flow to and from the RTCC.

The four levels of Processing are roughly arranged in descending order of priority. For example: if several similar messages are available for processing at the same time, P_1, which recognizes that a complete message is available, will always be carried out for all these messages first. When this is accomplished, processing will proceed to the lower levels of P_2, P_3, and P_4. Since P_1 represents the highest priority level of processing, and will be given preferential treatment in case of conflict, it was decided to first analyze the capabilities and limitations of the CP to perform this first level of processing. Therefore, this memorandum has been dedicated to the analysis of P_1. It is intended to directly follow this work with succeeding analyses to cover levels P_2 through P_4.

2. First Level Processing (P_1)

First level processing has been, until now, exclusively defined as a function, i.e., recognition that a complete data segment is in hand. Let us next consider how this function is accomplished with the Houston CP.

The CP system has been designed to operate in a "real-time" communications environment. As such it is required to rapidly respond to the communications needs of the various circuits interfacing with the CP. The method chosen for accomplishing this rapid response is the Program-Interrupt Technique. With this technique, the CP can go along performing useful work by executing a sequence of instructions in a program. When a communications interrupt occurs, the instruction sequence can be temporarily halted while the appropriate response to this "real-time" communications need, can be made.

There are two basic types of communications interrupts: external and internal. The external interrupt is activated by a signal announcing that a frame of data has come to its end. Therefore, since this analysis is limited to the recognition that a complete segment of data is in hand, we will be concerned only with external interrupts herein.

It should also be recognized that the end of a segment of teletype data is typically signaled by an internal interrupt. However, the considerations of teletype have not been included in this memorandum for two reasons:

First, teletype is relatively slow so that the time margins available for its handling are an order of magnitude greater than the high speed data. Secondly, the internal interrupts associated with teletype segments are handled at a lower priority than the external interrupts. Therefore, teletype interrupts can have essentially no effect on these loading considerations. Now that we have constrained the analysis to the recognition of external interrupts on high speed data lines, let us next investigate the way in which these interrupts are generated, and how they are handled.

a. External Interrupt Signals

The generation of external interrupts is shown pictorially on Figure 1. A data line, and a ready line* are shown connecting a peripheral receiver to a Standard Communications Subsystem (SCS). The SCS in turn is connected to the Univac-490 computer by an Input/Output (I/O) channel. The Communications Processor (CP) is comprised of the 490 computer and several SCS's, as well as several other peripheral units, not shown here.

*There are several other lines running between these two units; however, for this analysis, it is necessary to be concerned with only these two.

Let us now assume that a stream of data is coming into the peripheral receiver from a communications circuit. The data stream flows from the peripheral receiver to the SCS over the data line. From the SCS, the data flows into the 490 computer over the I/O channel.

Two frames of data (F_1 and F_2) are shown on the data line. Each frame is comprised of "M" characters, designated c_1, c_2, ... c_M.

It will be seen that a transistion from a negative to a positive state occurs on the ready line at the start of each data frame. While data is flowing, the ready line remains in the positive state. At the end of each frame there is a positive to negative transistion on the ready line. It is this negative transistion that signals an external interrupt, indicating the end of a frame of data. As shown on Figure 1, interrupt I_1 occurs at the end of frame F_1, and I_2 occurs at the end of F_2.

When an interrupt occurs, it is noted in the SCS by the setting of a logical element. This in turn causes an interrupt signal to be sent from the SCS to the 490 computer over the I/O channel. No further data transfers can occur on this channel until the interrupt is accepted by the computer.

The arrival of the interrupt signal in the 490 computer accomplishes two basic tasks. First, it causes a halt in the program in progress; and secondly, it initiates a jump to a special sequence of instructions which have been constructed to accommodate the interrupt. The operation of this instruction sequence will be covered in the next section.

b. Record Interrupt Sequence

The Record Interrupt Sequence is a short routine of approximately thirty-five instructions.* This sequence or process, which is essentially the heart of first level processing (P_1), requires about 300 microseconds (μs) for its execution time. If other interrupts occur after this sequence has started, they cannot be acknowledged until after the sequence is complete. Programwise, the Record Interrupt Sequence is given top preferential treatment in that another program sequence of instructions will always be halted while this Record sequence is executed.

After the Record Interrupt Sequence has been completed, a return is made to the Point Of Interrupt (POI). In other words, a return is normally made to the place in the program sequence that was in progress at the time of interrupt.

*This assumes an external interrupt on an incoming high speed data line, and that the program mode has been set to nonsuspendable.

The Record Interrupt Sequence performs four basic tasks. First, it must recognize that an interrupt has occurred. Secondly, it temporarily saves (stores) the required information concerning the segment of data which is now in storage as heralded by the interrupt. This information is required for subsequent processing. Thirdly, the Record Interrupt Sequence "toggles" specific high speed buffers. This is principally a task of address modification. It consists of supplying the address of the next buffer to be used. This "toggling" operation is necessary so that when a new frame of data begins flowing in on this particular circuit, it will be directed to the correct buffer area. Lastly, it is necessary to place the interrupt which indicates a received frame of data, in a queue. This interrupt queue is required because of the following set of circumstances. Each I/O channel is typically multiplexed, through a Standard Communication Subsystem (SCS), to serve several communication circuits. Since it is possible to have coincident interrupts on these several circuits, and since only one interrupt may be processed at a time, it is therefore necessary to establish an interrupt queue.

After the Record Interrupt Sequence has been completed, by performing the four basic tasks just described, it returns to the Point Of Interrupt (POI). The return to POI is caused by the last instruction in the sequence. In addition, this last instruction also causes release of the interrupt lockout. Therefore, if another interrupt had occurred in the interim, it can be accommodated at this time.

The foregoing discussion has described the operating characteristics that apply to first level processing (P_1) for messages received from the high speed data circuits. This type of traffic will encompass the bulk of the load which will be applied in this processing analysis. There is, however, one additional type of entry, which is the traffic from the Real Time Computer Complex (RTCC) at Houston. The operating characteristics pertaining to P_1, in the RTCC case, are essentially the same as those described for the high speed data circuits, with one notable exception. With the traffic from the high speed data circuits, a new buffer area is set up as part of the Record Interrupt Sequence. However, in the RTCC case, a new buffer area is not set up as part of the Record Interrupt Sequence, but is obtained a little later, under a lower priority worker program. This comes about because of the special nature of the RTCC demand-response interface. With this interface, data can be made to wait for an interval (up to several milliseconds) in the RTCC until the CP indicates that it is ready to receive it. Now that the basic characteristics of the first level processing function have been explored, they will be embodied in an algorithm which will be developed in the next section.

3. Processing Algorithm

A model of the first level Processing function (P_1) has been constructed in the form of an algorithm. This algorithm is a formalized logical description, in diagramatic form, that specifies the way in which P_1 operates.

The method of developing the model has been to rigorously specify the logical operations of the Communications Processor, when it performs the stated processing function, then to incorporate these logical operations into the form of an algorithm diagram. It is anticipated that this algorithm diagram will provide an analysis tool which may be applied to other spaceflight data loads of a similar type.

This section will be devoted to a general description of the algorithm, while a detailed description will be given in the Appendix A.

The basic task of the algorithm is to determine the amount of 490 computer time used by P_1 Processing; then to determine if this time usage had any adverse effects on overall Communications Processor operation. The most serious effect of consuming too much time in Processing, is the loss of data. This can occur if sufficient time is consumed by Processing so that no time is left for transferring a succeeding data character into the computer. In this case, the incoming data character would be lost. This, of course, points up the potential interaction between the two major CP functions of Processing and Transfers. It is therefore indicated that we must ultimately consider these two functions together.

The P_1 algorithm essentially operates by keeping track of a running variable "T", which indicates time. At some arbitrary time "zero", all of the potential entries for Processing are logged in. The first Processing entry is then chosen, according to an established priority scheme, and "T" is adjusted to reflect the amount of time consumed by Processing. After this, the next highest Processing entry is chosen, and "T" is again adjusted to reflect the P_1 Processing time used. This operation is iterative, continuing until all entries have been Processed.

As the algorithm is proceeding through its task of keeping track of "T", the status of succeeding data frames is being surveyed. This task is done in order to determine potential contention for the computer's resource of time. If contention is found, the amount of margin against data loss is measured. If the usage of this margin exceeds 100%, data loss results.

Now that the algorithm has been developed, the next step is to develop the processing loads that will be applied to this algorithm.

4. The Processing Load

The objective here, is to develop a processing load to apply to the algorithm. It is desirable for this load to apply the maximum stress that could actually occur during a Gemini mission. Therefore, let us first develop the type of load that produces a stress situation, then develop the actual loads.

a. A Stress Situation

As stated earlier, once processing (P_1) has started on a selected interrupt, all other interrupts are ignored (locked out), until the end of the P_1 sequence.

This then, constrains the 490 computer, to processing one interrupt at a time. Therefore, because of this one-at-a-time characteristic, an element of a stress situation, is to have a relatively large number of interrupts occur coincidently.

When an interrupt occurs on some I/O channel, the succeeding frame of data on this channel cannot be accepted into the 490 computer until this interrupt is acknowledged.* Therefore, the other element of a stress situation, is to have the succeeding frame of data follow very closely, the frame that caused the interrupt. An attempt has been made to maximize these two characteristics in the selected load.

b. A Stress Load

Stress loads have been previously developed, in collaboration with the NASA-MSC communications group, for the Transfer[1] and Storage[2] analyses. These loads were based upon a Gemini two-vehicle mission. The time period assumed occurs several minutes after the Gemini launching with the Gemini vehicle in radar view of both Cape Kennedy and Bermuda. In addition, an Agena vehicle, which was previously launched, is on a Cape Kennedy flyby.

*The interrupt is acknowledged by executing a "Store Channel" instruction.

The same type of load has been found suitable for this processing analysis, with many of the entries that were used in the other analyses carried over. However, in order to maximize the stress characteristics just developed, it was assumed that frame endings were coincident. This was not the case in the "Transfer" analysis loads, where frame endings were non-coincident.

Each line on Figure 2 represents an external interrupt, which has heralded the end of a frame of data or message; the lines being in descending order of priority. All of the basic information concerning each interrupt is given on the figure. This includes the identity of the I/O channel, and Communication Line Terminal (CLT) contributing the interrupt; the number of bits in each frame, the number of frames per second, the time taken to transmit each frame in milliseconds, transmission rate on each communication line, as well as the source of each data frame.

It will be noted that this represents the stress "live" load only. In addition, a "simulation" load has been developed, in the same type of format, and is shown on Figure 3.

5. Derivation of Results

In order to more clearly demonstrate the derivation of the analysis results, it was decided to construct a graphical presentation. This has been carried out for the Gemini live mission alone, and then another graph depicting the results of combining a Gemini "simulation" load with the "live" mission, has been constructed.

The graphs are in the form of time usage charts, which display the amount of time used for the P_1 Processing operation.

a. Gemini Live Mission

The results of processing the Gemini live load, have been plotted graphically on Figure 4. The end of each data frame, which causes an interrupt, is shown on the left-hand side. Each frame is identified as to type of data and the number of the Communication Line Terminal (CLT) contributing the data frame. The time consumed by the Processing operation (P_1) is shown by the dashed line step function on the figure.

The interrupting frames are arranged in descending order of priority and are grouped by I/O channels. The first frame is the teletype broadcast coming from the Real Time Computer Complex (RTCC), over I/O Channel 14. It will be seen that it requires about 0.13 ms to process this frame, i.e., record the interrupt. Since this is the only interrupt on Channel 14, processing next moves to Channel 13.

The first frame to be processed on Channel 13 is the Gemini telemetry coming into the 490 computer from CLT Number 73. It will be seen that this frame had to wait for 0.13 ms while the teletype broadcast frame was being processed. Next a frame of Bermuda radar is processed and so on down the Channel 13 interrupts, until processing has been completed on this channel. This occurs at 1.93 ms, as shown on the drawing.

The processing operation now moves down to Channel 12, where it first serves the GE/Burroughs frame of data. Again, processing continues down to the Cape Kennedy Impact Predictor frame, where it is completed at 3.13 ms.

In the development of this Processing analysis, it has been revealed that the computer's resouce of time is subjected to stress by P_1 Processing. Therefore, the heavier the P_1 load, the more computer time will be used. Exceeding the P_1 time limitation is not reflected in degradation to P_1 itself, since P_1 is the highest priority task, and will always be performed first. Rather, if P_1

uses excess time, insufficient time will remain for the computer to perform its remaining tasks. If this happens, the most serious outcome envisioned is one that would result in lost data.*

Lost data could occur if the 490 computer had used sufficient time in carrying out P_1 so that no time remained to perform character Transfers. In order to investigate this possibility, the character Transfer request, occurring at the shortest possible time after the coincident interrupts, was chosen. This is a request to transfer the first character in a succeeding frame of TITAN, as shown in Figure 4. It should be noted that the TITAN TLM frame on Figure 4 is the only one concerned with character transfer; therefore, this frame is not as yet in the computer. The other frames shown are already in the computer, and are being P_1 Processed.

It will be noted that the frame of TITAN telemetry started to come into CLT-47 at 0.4 ms. This is the time at which the first bit of the first character starts serially into the assembly register of the CLT. The serial bit stream comprising this character will be completely in the assembly register of the CLT by about 2.8 ms. At this time,

*There is also a less serious outcome envisioned which is the potential delay to character transfers from the RTCC. This is covered in Appendix B.

the character will be ready for parallel bit transfer into the 490 computer. Its readiness for transfer is signaled by a transfer request, as indicated by the upward pointing arrow at about 2.8 ms. If the computer is free (not performing a higher priority task) to accept the character at this time, it will be immediately transferred in. In the case of the live mission load alone, as shown on Figure 4, the computer is free at this time. It is free because all higher priority requests (interrupts) have already been processed. Therefore, this first character in the TITAN telemetry frame is transferred into the 490 computer immediately.

If the 490 computer had not been free to transfer this character immediately, it would wait in a queuing register in its CLT. Since it is a single character queuing register, the character may wait only for the length of time it takes to assemble the next incoming serial character. The limit on waiting time in this specific case is shown to be 5.2 ms (Figure 4). If this limit is exceeded, data loss will result. However, this is of no concern here, since the character was transferred immediately, and did not have to wait in the queuing register.

It has been shown that the request to Transfer the first character in the TITAN frame occurs at 2.8 ms. Further, this request must be serviced before exceeding the time limit of 5.2 ms. Therefore, this interval between 2.8 ms and 5.2 ms may be considered as the margin against data loss. As noted earlier, none of this margin against data loss was used by P_1 Processing the live load.

It should be realized, before leaving this section, that other data frames also start in the time interval encompassed by the abscissa. However, the most critical case envisioned, was the TITAN frame shown. Therefore, the other frames occurring in this interval are not shown.

b. Gemini Combined Live and Simulated Missions

The results of processing the combined loads have been plotted graphically on Figure 5. The small letter "s" on several frames indicates that they are simulation entries. The results are interpreted in the same way as the live mission (only) just covered in the previous section. However, there are several differences which require some discussion.

A Delta Clock interrupt has been added to the Processing load. The Delta Clock is the only function that has a higher priority than the external interrupts. Therefore, with an assumed Delta Clock interrupt occurring at about 2.5 ms, it would capture a servicing interval between Channels 13 and 12 as shown in Figure 5.

In addition to the Delta Clock interrupt, it has also been assumed that a request to transfer a character out to the magnetic tape unit has occurred at about 4 ms. This would not cause any difficulty in that it could only add, at most, two 4.8 µsec. memory cycles to the processing time used. This, of course, is so relatively small that it cannot be seen on the millisecond scale on Figure 5. Also, a small delay in transferring a character to tape will have no adverse character transfer consequences as discussed in the Transfer Memorandum. In addition to this output transfer, several incoming character transfers could also occur in this interval, with no significant effect.

The combining of live and simulation loads has necessarily placed a heavier burden on processing time. Therefore, the request at 2.8 ms to transfer the TITAN character cannot be served immediately, resulting in the use of some of the margin against data loss. It will be noted on Figure 5 that at the time the transfer request comes up (2.8 ms) on Channel 12, the 490 computer has several higher priority interrupt requests as yet unserved on this channel. In other words, the computer must acknowledge the outstanding interrupts from CLT's 63, 53, and 47 on I/O Channel 12 before the TITAN request can be served. This occurs at about 3.6 ms as shown on Figure 5, as the start of the remaining margin. Now taking the range between 2.8 ms and 5.2 ms as the total margin against data loss, it will be noted that we have used about 33% of this margin, i.e., $[{}^{(3.6-2.8)}/_{(5.2-2.8)}] \times 100$.

6. Results

The P_1 Processing capabilities of the Univac-490 Communications Processor were shown to be adequate to handle the Gemini peak loads developed in Section 4. This has been shown on the time usage charts of Figures 4 and 5.

When the peak Gemini live load was applied to the algorithm, there was no reduction in the margin against data loss. Next, the simulation load was added to the live load. With this combined load, the time usage of Processing was such that, in the worst case, about 33% of the margin against data loss was used.

It has been established earlier that the most serious consequence of using an excess amount of time in P_1 Processing is potential data loss. In some cases, the likelihood of lost data is greater than others. Specifically, data loss is more likely when there are very short time gaps between successive frames of data. With short gaps the P_1 Processing time at the end of one frame may be very close or even overlap the first character transfer associated with the start of the next frame. Therefore, in order to apply the most critical criterion, in determining the margin against data loss, a circuit was chosen with the shortest gaps between frames of data. To cause even more stress, the circuit is low in the order of service priority. Specifically, it is a line that carries the incoming Gemini and TITAN telemetry data frames into the Communications Processor. The 33% margin usage, cited in the previous paragraph, applies to this critical line only. Other lines use considerably less margin.

The significance of exceeding the margin is that succeeding characters of data, seeking entry into the CP, cannot be accommodated because the Processing time usage has consumed the margin. When these characters cannot be accommodated in time, they remain in a register too long and are overwritten by the next succeeding character.

A. E. Peterson

WH-4332-AEP-RP　　A. E. PETERSON

Att.
Appendices A and B
Figures 1-5

Copy (with attachments)to
See next page

Copy (with attachments) to
Messrs. J. J. Hibbert - Bellcomm
C. R. Moster - Bellcomm (50)

C. A. Armstrong - WH
P. V. Dimock - WH
R. M. Marella - WH
R. J. McCune - WH
C. W. Schramm - WH
M. P. Wilson - WH

REFERENCES

1. Project Apollo - Communications Processor - Analysis of Character Transfer Function - Case 20061
   Dated March 31, 1965, From: A. E. Peterson and C. W. Schramm, MF5-4332-11

2. Project Apollo - Communications Processor - Analysis of Storage Function for Data Flow to the MOC/DSC - Case 20061
   Dated June 16, 1965, From: R. M. Marella, MF5-4332-31

EXTERNAL INTERRUPTS

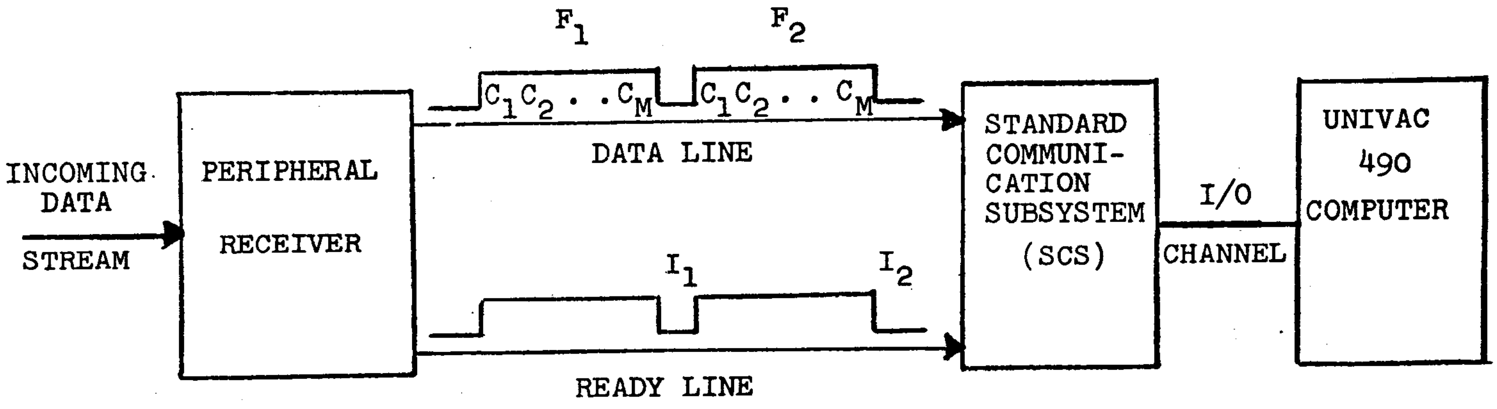


FIGURE 1

THE GEMINI LIVE LOAD

I/O CHAN	CLT PRIORITY NO.	TYPE OF DATA FRAME	BITS PER FRAME	FRAMES PER SECOND	TIME PER FRAME IN MS	XMSN. RATE KB/SEC.	FRAME SOURCE
14	--	TTY BRDCST.	2268	--	55.6	40.8	MOC
13	73	GEMINI TLM	1740	1	362.5	4.8	B/F CH. 3
13	73	TITAN TLM	432	1	90.0	4.8	B/F CH. 3
13	57	BDA RAD	192	10	96.0	2.0	DCU(R)-1
13	53	CKEN IP	480	4	240.0	2.0	DCU(R)-1
13	47	GE/B	480	2	240.0	2.0	DCU(R)-1
13	23	CMD-RTS AMR	65	--	65.0	1.0	MDCS-1
13	13	CMD-RTS BDA	65	--	65.0	1.0	MDCS-1
12	73	GE/B	480	2	240.0	2.0	DCU(R)-2
12	47	GEMINI TLM	1740	1	362.5	4.8	B/F CH. 3
12	47	TITAN TLM	432	1	90.0	4.8	B/F CH. 3
12	13	BDA RAD	192	10	96.0	2.0	DCU(R)-2
12	03	CKEN IP	480	4	240.0	2.0	DCU(R)-2

FIGURE 2

THE GEMINI SIMULATION (SCATS) LOAD

I/O CHAN	CLT PRIORITY NO.	TYPE OF DATA FRAME	BITS PER FRAME	FRAMES PER SECOND	TIME PER FRAME IN MS	XMSN. RATE KB/SEC.	FRAME SOURCE
14	--	CMD LOAD	180	--	4.4	40.8	SOC
14	--	AGENA TLM #3	324	1	8.0	40.8	GSSC
13	37	SCATS CMD	65	--	65.0	1.0	SDCS-1
12	63	SIM "CMD LOAD"	96	--	96.0	1.0	MDCS-2
12	53	SIM "RTS"- CMD	65	--	65.0	1.0	MDCS-2

FIGURE 3

TIME USAGE CHART

PROCESSING (P_1) THE LIVE GEMINI LOAD

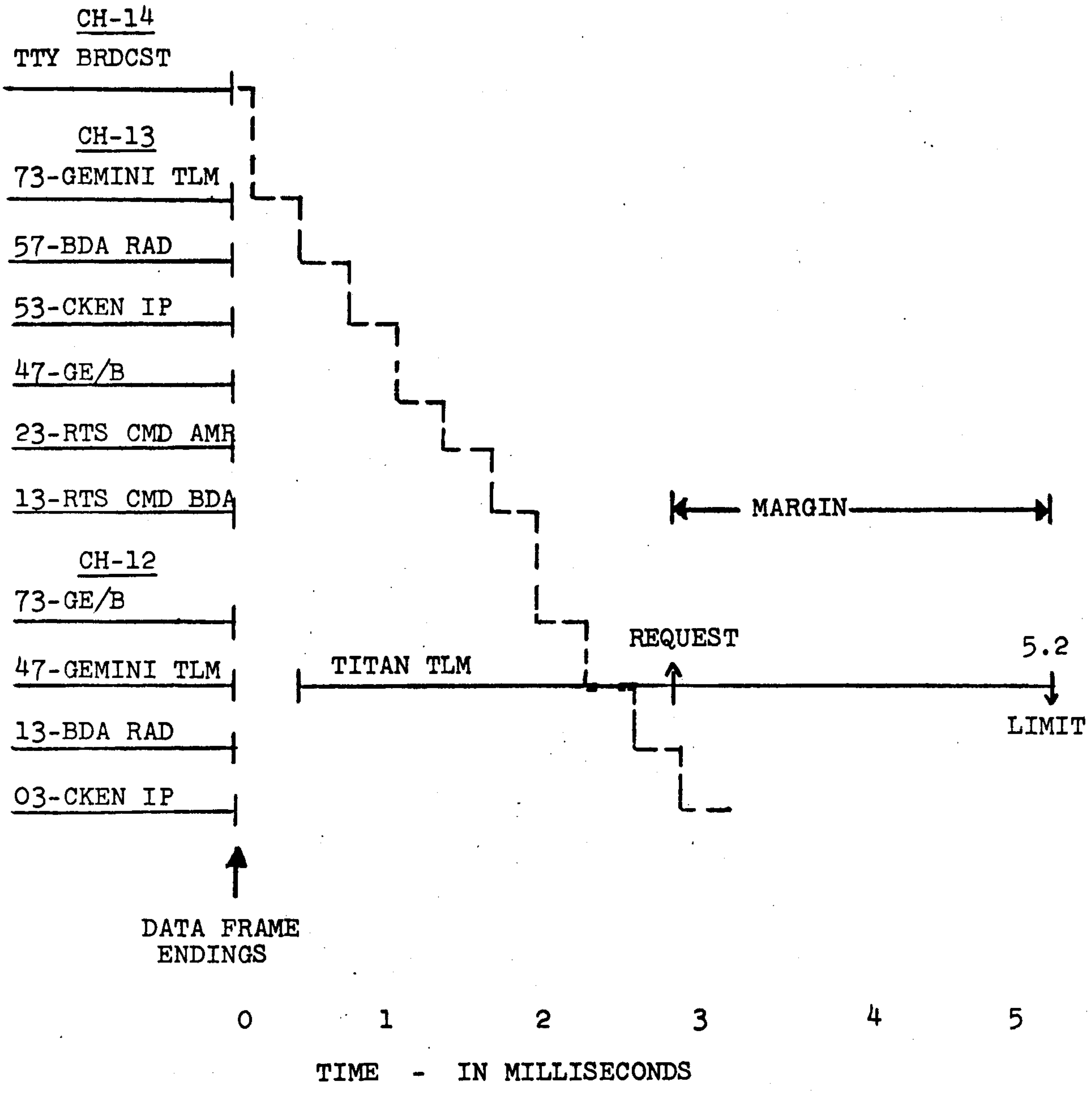


FIGURE 4

ISSUE	ENGR	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED	
	DRAWN		NO. OF SHEETS PER SET	SHEET

PROCESSING (P_1) THE COMBINED GEMINI LOAD

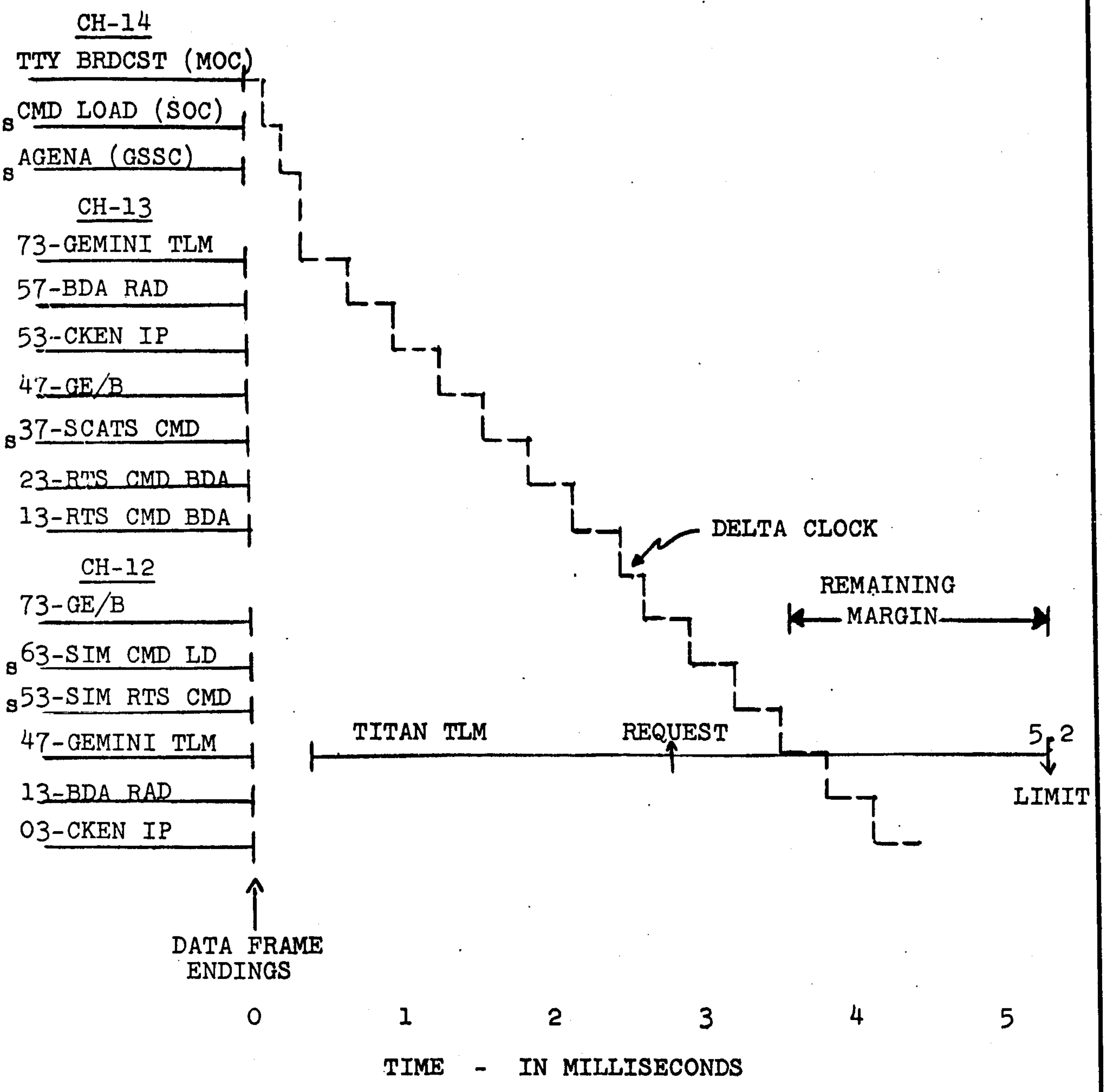


FIGURE 5

APPENDIX A

The P_1 Processing Algorithm

General

The P_1 Processing algorithm is a logical model, in diagramtic form. It specifies the operational flow of the P_1 Processing function performed by the UNIVAC-490 Communications Processor (CP). The task of P_1, is to recognize that a frame of data has been delivered into the CP. This task is essentially accomplished by a Record Interrupt Sequence of instructions. This sequence is executed when an incoming data frame has been completely received.

The basic function of the algorithm is to determine the amount of 490 computer time used by P_1 Processing. In addition, this time usage is evaluated, to see if it has exceeded some limit. The time limit for P_1 usage is the point at which serious degradation occurs in Communications Processor performance. The most serious degradation envisioned, is the loss of data. This could occur if enough time was used by P_1, so that insufficient time remained to transfer an incoming data character into the 490 computer. Therefore, the time status of incoming data transfer requests is periodically surveyed and compared with the time status of P_1 Processing.

In this way, it is determined, whether or not P_1 time usage has exceeded, or even approached its time limit. This is the main output of the algorithm. The criterion chosen to measure this time usage, is the margin against data loss. The margin is a range in time. The start time of the margin, is the time at which a data character is available for transfer into the computer. If the computer is free at this time, i.e., performing no other higher priority tasks, it can transfer this data character immediately. In this case, none of the margin would be used. The terminating time of the margin, is the time limit for transferring this data character into the computer. If the character is not transferred by this time, it will be destroyed by the next incoming character. These two extremes are taken as the zero, and one-hundred percent usage points respectively, of the margin against data loss. The algorithm also permits determination of all percentage points between the two extremes, as well as points exceeding one-hundred percent.

A subsidiary output of the algorithm is the amount of delay to the transfer of incoming characters from the Real Time Computer Complex. This potential delay comes about because of the demand-response nature of the interface between the Communications Processor (CP), and the Real Time Computer

Complex (RTCC). With this interface, data characters from the RTCC will not be accepted by the CP while the CP is performing its P_1 Processing function. Therefore, these data characters, if there are any at this time, must be held in the RTCC (delayed) until the CP has completed P_1 Processing.

Although the algorithm is not limited to the specific situation used in this analysis, it is constrained to accomodate this type of situation. Specifically, the algorithm is designed to handle a P_1 Processing load that consists of all external interrupts, and perhaps one or two Delta clock interrupts. The load is expected to have coincident frame endings, which will be on circuits that are multiplexed onto three 490 computer I/O channels. One channel will be used to serve the Real Time Computer Complex (RTCC) circuits, and the other two I/O channels will serve all the high speed data circuits.

It has been convenient to break the algorithm into a series of sub-functions, and display each sub-function on a separate diagram. These are shown as diagrams 1 through 9. Preceding the diagrams is Table 1, which lists the various symbols used throughout the diagrams. The various shaped enclosures in the diagrams are used to denote the different operations, which are performed in the algorithmic process.

The rectangles represent actions which are taken, while the ovals represent arithmetic manipulations. The diamonds take care of the logic, in the form of tests and binary (yes-no) decisions. The numbered circles are connection points, with identical numbers denoting common points in the process.

1. Initialization

The first sub-function in the algorithm is Initialization, and is shown on Diagram 1. It is here, that the starting conditions, for the algorithm, are established.

a. Setting "T"

The first operation, is of the arithmetic type, which sets the running variable "T" to zero. "T", of course, is primarily used to keep track of the amount of 490 computer time used by P_1 Processing.

b. Logging External Interrupts

The next operation, which is an action, logs in the external interrupts at T = 0. These interrupts represent the P_1 Processing load, because each interrupt signifies that a frame of data has been completely delivered to the 490 computer. It will be noted that the algorithm has been designed to handle a group of coincident frames, i.e., each frame is completely in the 490 computer at T = 0. The reason for choosing coincident frame endings is that this situation places the greatest stress on P_1 Processing.

c. Logging Delta Clock Interrupts

The next rectangle provide the opportunity to introduce a Delta Clock interrupt. Since the Delta Clock interrupt is the only higher priority function, than External interrupts, it must be accomodated when it occurs.

d. Logging Transfer Requests

The next operation is concerned with logging in the initiation times of Transfer requests (R_{ij}). This is required because the algorithm must determine a P_1 Processing time limit, which is a function of character transfer times.

Input data request (for transfer) times may be determined from the following equation:

$$R_{ij} = F_{ij} + (B_{ij}\ C_{ij}) - B_{ij}/2$$

where

R_{ij} = Request initiation time to transfer the first character of the first frame following the interrupt, on the ith circuit of I/O channel j,

F_{ij} = Frame start time, i.e., the time at which the first bit, of the first character, of the next frame (following the interrupt on the ith circuit on I/O channel j), enters the assembly register in the CLT,

B_{ij} = Time required to transmit each bit on the ith circuit on I/O channel j, and

C_{ij} = Character size, measured in the number of bits, on the ith circuit on I/O channel j.

For example, the request (R_{ij}) to transfer the first character in the next frame of data, following the interrupt on circuit 47 (CLT priority number), on I/O channel 12, is computed as follows:

This is a frame of TITAN data. It starts into the CLT at 0.4 ms (after time zero); therefore, $F_{ij} = 0.4$. Since this circuit operates at a transmission rate of 4.8 kB/second, each bit required 0.208 milliseconds (ms) for transmission or $B_{ij} = 0.208$ ms. Finally, each character in this TITAN frame is comprised of 12 bits; therefore, $C_{ij} = 12$. Now putting the whole equation together, we have $R_{ij} = 0.4 + (0.208 \times 12) - 0.208/2 = 2.792$ ms.

In addition to input data requests for Transfers, the algorithm is equipped to handle requests to transfer characters out to the high speed data circuits. It has not been designed to accomodate output requests for character transfers to the RTCC, since these are not expected to present a problem (see Appendix B).

Output high speed data circuits requests may be introduced at times when they would be expected to occur. It is not envisioned that any computation will need to be performed to introduce these output requests. Rather, a value, or values of R_{ij} may be postulated and introduced directly. However, in logging requests, it is necessary to differentiate the outputs from the inputs, because the algorithm handles them differently.

e. Logging Transfer Limits

The next operation in Initialization, is the logging of the time limit for serving each input request. The emphasis herein is on input requests, because a failure to serve an input request in time, results in data loss. On the other hand if an output request is not served promptly, it is merely delayed.

All of the input (to the CP) line terminating units (CLT's and CCU's) are equipped with single character queuing registers. When an incoming character is assembled, it is placed in this queuing register. At the same time the transfer request signal (R_{ij}) is generated. If the 490 computer is not performing another task at this time, the character will be transferred in immediately. If the computer cannot serve the request immediately, the character waits in the queuing register. Since it is a single character queuing register, the waiting character may remain in the queuing register, only for the amount of time it takes to assemble the next character. This is given as (B_{ij} C_{ij}), where B_{ij} and C_{ij} are the bit times, and character lengths, as described previously. It is necessary, to reference the time limit for the character wait, to the time when it is placed in the queuing register (R_{ij}). Therefore, the time limit for serving each input data request for character transfer is given as $L_{ij} = R_{ij} + (B_{ij}C_{ij})$. For example, the time limit for serving the TITAN frame coming in on CLT-47 of I/O channel 12 is $L_{ij} = 2.792 + (0.208 \times 12) = 5.288$ ms.

Now that all the necessary information has been logged in or recorded, we move on to the selection of the first interrupt for P_1 Processing.

f. Selecting Interrupt

Selection of the highest priority interrupt is essentially conducted at two levels. The I/O channel is the major priority, and the communication circuit is the minor priority level. The highest numbered channel is given precedence over the lower numbered channels. Therefore, an interrupt on channel 14 would be served before an interrupt on channel 13, etc.

When an I/O channel has been selected, the particular circuit on that channel to be granted service, depends upon the priority number of the CLT that terminates the circuit. Again, the highest numbered CLT will be given preference over the lower.

On I/O channel 14, which interfaces with the RTCC, the Communications Control Unit takes the place of the CLT. In this case, the priority of the circuits from the RTCC from high to low is: MOC, SOC, and GSSC.

2. Processing Interrupts

The second sub-function in the algorithm is P_1 Processing as shown on Diagram 2. This, of course, is the heart of the algorithm, where we keep track of the amount of time consumed by P_1 Processing.

It will be seen that this diagram is entered through connector 1. This is the point from which we have just come, on the previous diagram (1). Therefore, the first interrupt to be processed has been selected. Now it must be determined, whether or not this external interrupt is from a high speed data (HSD) circuit (line). If the interrupt is from an HSD line, it will require about 0.30 ms for Processing. This is shown on Diagram 2 by increasing the value of "T", which is initially zero, by 0.30 ms. This is the amount of 490 computer time consumed by going through the "Record Interrupt Sequence" of program instructions, as well as "Toggling" the high speed buffers.

If the interrupt is not from an HSD circuit (line), it will be from the RTCC or the Delta Clock. It is recognized that other types of interrupts are possible in this system, however, this algorithm is constrained to the external HSD, and RTCC interrupts generated by incoming data frames, as well as the Delta Clock interrupts. In either case (RTCC or Delta Clock), only 0.13 ms is required for P_1 Processing, which is shown in the lower oval on Diagram 2. This reduction in Processing Time (0.30 ms to 0.13 ms) comes about because there is no "Toggling" operation to be performed when "Recording" an RTCC or Delta Clock interrupt.

After the value of "T" is properly adjusted, the algorithm is at the connector 2 point. From here, the process moves on to connector 2 on the next sub-function shown on Diagram 3.

3. Selecting the Next Interrupt

The sub-function of selecting the succeeding interrupt for Processing, is shown on Diagram 3. The first opertion is a step of logic. It is used to determine whether or not there are more interrupts (data frames) for P_1 Processing. If the answer is yes, the highest priority outstanding interrupt is selected for Processing. However, before moving on to Processing, another determination is made as shown by the second diamond enclosure on Diagram 3. Here it is determined whether or not there is a possibility of using any of the margin against data loss. If all requests for transfers are to be released* at times greater than "T". (T reflects the current time status of P_1 Processing), there can be no contention between transfer and Processing. It therefore follows, that there can then be no usage of the margin against data loss. If there is no possible contention, a "yes" answer results here. This, of course, places the process at connector 1, from where it is returned to Diagram 1. This permits the last selected interrupt to be Processed.

If a "no" answer had resulted at the second decision point on Diagram 3, it would have indicated the possibility of some margin usage. In this case the process would have been directed to connector 3 on Diagram 3, which leads to the next diagram.

* Transfer requests are released at their initiation times (R_{1j}), which were logged in as part of the first sub-function of initialization.

4. Classifying Transfer Request

The process of classifying a transfer request is shown on Diagram 4. On entering this diagram, it has already been established, that one or more requests for transfer occur at times, that could conflict with P_1 Processing. The basic purpose of this sub-function, is to deal with these requests systematically.

It should be recognized that there may be more than one transfer request to deal with. Therefore, if several requests have been initiated, we must somehow choose them, one-at-a-time, and examine them. The mechanism for choosing them, is on the basis of priority. It has been decided to select the highest priority request first, since this would typically be the first one to be served. This operation is shown by the rectangle on Diagram 4.

After the request is selected, it must be classified, since different types of requests are handled differently by the algorithm. At the first step in logic (the diamond enclosure), it is determined whether or not it is an RTCC request. Assume for the moment, that it is not an RTCC request. In this case, it must be a high speed data (HSD) line request, and we obtain a "no" answer. This takes us to the second decision point on Diagram 4. Here, it is determined if the transfer request is on the other HSD I/O channel of the 490 computer, which is not active with the current interrupt. If this is the case, a "yes" answer is obtained, and the process moves to the next decision point. Here it is determined whether or not the request priority is greater than the interrupt priority. If the answer is yes, the process moves on to the next diagram, through connector 4, where the request will be Processed. To recap; this chain of logic has classified this transfer request as an HSD line request, that has occurred on the higher priority HSD I/O channel. At the same time, it should be realized that the currently selected interrupt, is on the lower priority HSD I/O channel.

If the request priority had not been greater than the interrupt priority, it would have indicated that the request was on the lower HSD I/O channel. In this case, the request could not be served, because of the outstanding interrupts on this lower priority channel. Therefore, the request is inhibited for one P_1 service interval before it is again examined. The reason for inhibiting the request for one service interval is twofold. First, the situation may change after the next service interval, i.e., the request and interrupt could now be on the same I/O channel. Secondly, we want to remove this request from the next test, i.e., "More $R_{ij} < T$"? This test determines whether or not there are any more transfer requests to be examined. If the answer is yes, the process is returned to the top of Diagram 4, through connector 3, and a next request is selected, etc. If the answer is no, we return to Diagram 2, through connector 1, and Process the last selected interrupt.

If the request had been on the same channel as the interrupt, and this was the lower priority channel, the process would have led to connector 7 on Diagram 4. This leads to Diagram 8 which is described later. If both request and interrupt had been on the higher priority channel, it might be necessary to jump down to the lower I/O channel to Process the next interrupt. This will occur if the request is from a higher priority CLT than the currently selected interrupt. The reason for this is the outstanding interrupts on the lower priority HSD I/O channel, which are a higher function priority than transfer requests. This logic is shown by the lowest diamond and rectangle in the center of Diagram 4.

5. Serving HSD Request on a Non-P_1 Busy Channel

The serving of a high speed data request on a non-P_1 busy channel* is a simple process as shown on Diagram 5. In

*A channel that is not engaged in the recording of external interrupts.

serving the request (effecting the transfer), a small amount of computer time is used. The time used amounts to; two 4.8 microsecond (μs) memory cycles, or a total of 9.6 μs (0.0096 ms), as shown in the oval. One memory cycle is used to up-data a Buffer Control Word (BCW), and the other to actually transfer the data character. This operation is not limited to input or output transfers, but will handle either type. After the request is served, it must be determined whether or not there are any more requests that have been released, i.e., any $R_{1j} < T$. If the answer is yes, the process returns through connector 3, to Diagram 4. Here, the request goes through the classifying sub-function, which has been previously described. If the answer is no, on Diagram 5, the process is returned to connector 1 on Diagram 5. Then it returns to connector 1 on Diagram 2 for Processing the interrupt.

It will be noted that no record is kept, of the delays in serving transfers on this non-P_1 busy channel. The reason for this is as follows; the operating characteristics of the Communications Processor are such, that a significant delay to these transfers is virtually impossible. In other words, transfer requests on the higher priority non-P_1 busy channel will always be served almost immediately.

6. Logging RTCC Requests

When a request to transfer a data character is not associated with a high speed data (HSD) line, it is associated with the Real Time Computer Complex (RTCC) lines. In this case (RTCC), the transfer request is handled differently than the HSD line requests. The algorithm sub-functions on Diagrams 6 and 7, are dedicated to handling these RTCC data transfer requests.

Diagram 6 is entered from connector 5 on Diagram 4. At this point it has been established that the transfer request is not associated with a high speed data line. Therefore, it must be associated with the RTCC lines.

It will be noted that the first operation on Diagram 6, is to log the RTCC request. The reason for logging, and not serving the request, is dictated by the demand-response characteristics of the interface between the Communications Processor (CP) and the RTCC. This interface operates with a "Ready-to-Receive" line. This line is activated by the CP when it is ready to receive data characters from the RTCC. Activation of the "Ready-to-Receive" line takes place in a sequence of instructions contained in a communications worker program. However, as long as the CP is conducting the high priority P_1 Processing function, it will not get to the worker program that activates the "Ready-to-Receive" line. Therefore, any RTCC data transfer requests are logged, and served only after all P_1 Processing is completed.

After the RTCC Transfer request is logged, as shown in the rectangle of Diagram 6, it is next determined if any more request have been initiated before time "T". Depending on the outcome, a return will then be made to either Diagram 3 or 4, as described in the previous section.

7. Serving RTCC Request

After all the P_1 Processing is completed, the Ready-to-Receive line to the RTCC, may be activated. Then, data transfers may be made between the RTCC, and the CP. The sub-function shown on Diagram 7, is devoted to effecting these transfers.

This diagram is entered from connector 6 on Diagram 3. At this point, it has been determined that all of the interrupts have been Processed. Therefore, we are now ready to proceed with the RTCC transfers.

The first step is one of logic. It determines if there had been any RTCC Transfer requests. If there are requests in the log, the highest priority one will be selected first, since this would be the first one to be served. Next the character is transfered, which uses 0.0096 ms of time.

The next operation is a computation of the amount of time the request had to wait (delay) before being served. Since the request was initiated at R_{1j}, and it wasn't serviced until "T", it suffered a delay of $(T-R_{1j})$. Actually, this delay quantity of $(T-R_{1j})$ is not rigorous, since the request would not be served exactly at time T. Rather, it would be served a little later than T, because several instructions would have to be executed before the Ready-to-Receive line could be activated. However, the quantity $(T-R_{1j})$ is approximately correct, in that the error is considered insignificant.

The next operation is merely a recording of the amount of delay. It seems advisable to keep a record of these transfer delays for a given load situation, since it is conceivable that their magnitudes could be serious.

The sub-function is now returned to connector 6, from where it can be repeated as often as necessary. When all RTCC requests have been served, the algorithm is terminated as shown.

8. Serving a High Speed Data Transfer Request on a P_1 Busy Channel

This sub-function, which is shown on Diagram 8, delineates serving another type of character transfer request. This is a high speed data (HSD) line request on an I/O channel, that is busy with P_1 Processing. Serving this type of request is potentially the most critical operation included in the algorithm. For if an inordinate amount of time has been used for Processing, not enough time will remain to service this type of request, and lost data will be the result.

The first operation on this diagram is entered from Diagram 4, connector 7. At this point, it has been established that the transfer request is on the same 490 computer I/O channel as the currently selected interrupt. In addition, it has been established on Diagram 4, that both request, and interrupt are on the lower priority HSD Computer I/O channel. Now, if the circuit (CLT) priority of the Transfer request is lower than the interrupt priority, the request will not be served. In this case, this request will be inhibited for one P_1 Processing interval before it can be examined again.

If there are no more requests with initiation times less than "T" ($R_{1j} < T$), the process returns through connector 1 to Diagram 2, where the most recently selected interrupt is Processed. If there are more requests with initiation times less than "T", they are classified by returning to Diagram 4, through connector 3. Then they are Processed, as described earlier in the discussion of Diagram 4.

When the priority of the Transfer request is equal to or greater than the interrupt priority, it is served by increasing "T" by 0.0096 as shown in the oval on Diagram 8. Although, in actual practice the interrupt would be served first, the request is served first in the algorithm. This is simpler, and it is not considered to introduce a significant error in the algorithm. The only time at which this could be significant, is when the margin used, is calculated. Therefore, at this time, the time differential introduced by this inverted service ordering, is corrected.

If the request just Processed, in the oval of Diagram 8, is an "output" (from the CP) request, there can be no data loss, because the data is held in the CP until it can be outputted. In this case, the process returns to the upper branch and continues on normally. It should be noted that any delays to these output character transfers have not been computed. The reason for this is that they are not expected to be significant. However, in the future it may be desired to expand the algorithm to take account of these delays. In this case, delays may be simply computed, in the same way as the RTCC requests, which were explained in Section 7.

Let us now assume, that the transfer request served in the oval on Diagram 8 was an "input". Since input transfers must be accomodated in time, or lost data will result, the amount of margin used in serving this request must be computed. This computation is covered on the next diagram (9).

9. Computing Margin Used

This sub-function, which is shown on Diagram 9, is dedicated to determining just "how close" we have come to the time limitation of P_1 Processing. This limit, as set forth earlier, is the point at which data loss occurs.

At the outset (connector 8), it has been established that a data character has been transferred into the 490 computer. The time at which it will actually be in the computer is (T + 0.05 ms). The derivation of this time is covered in the following paragraph.

The value of "T" here, is the time at which P_1 Processing has been completed for the previous interrupt, plus the time to transfer this character. However, this character transfer cannot take place until after a "Store Channel" instruction is executed. This Store Channel instruction will be executed about 0.05 ms after Processing has commenced on the current interrupt. Therefore, the time at which the data character will be transferred into the computer is (T + 0.05).*

* This corrects the inverted order of servicing, which was discussed in the previous section.

The time range or margin, over which the character Transfer may take place, is the limit time less the request initiation time, or $(L_{ij}-R_{ij}{}^{*})$. The segment of this margin that was used to effect the transfer is $[(T + 0.05) - \cdot R_{ij}]$. The percentage of margin against data loss used, is given by their ratio, times 100, as shown in the oval on Diagram 9. If this quantity exceeds 100%, data loss is indicated.

Finally, the margin used in this particular case (the ith circuit on channel $_j$) is recorded, and the algorithmic process continues. At the decision point we return to connector 1 on Diagram 2, or connector 3 on Diagram 4, depending upon the presence, or lack of more request initiations.

* The derivation of these quantities is covered under "Initialization in Section 1.

TABLE 1

TABLE OF SYMBOLOGY FOR ALGORITHM

Variables

T = Running Variable for Time - Its value depends on its sequential position in the algorithm.

R = Initiation Time of a Request to transfer a data character in or out of the central computer of the Communications Processor.

L = Time Limit for serving a Request to transfer a data character into the central computer.

Indices

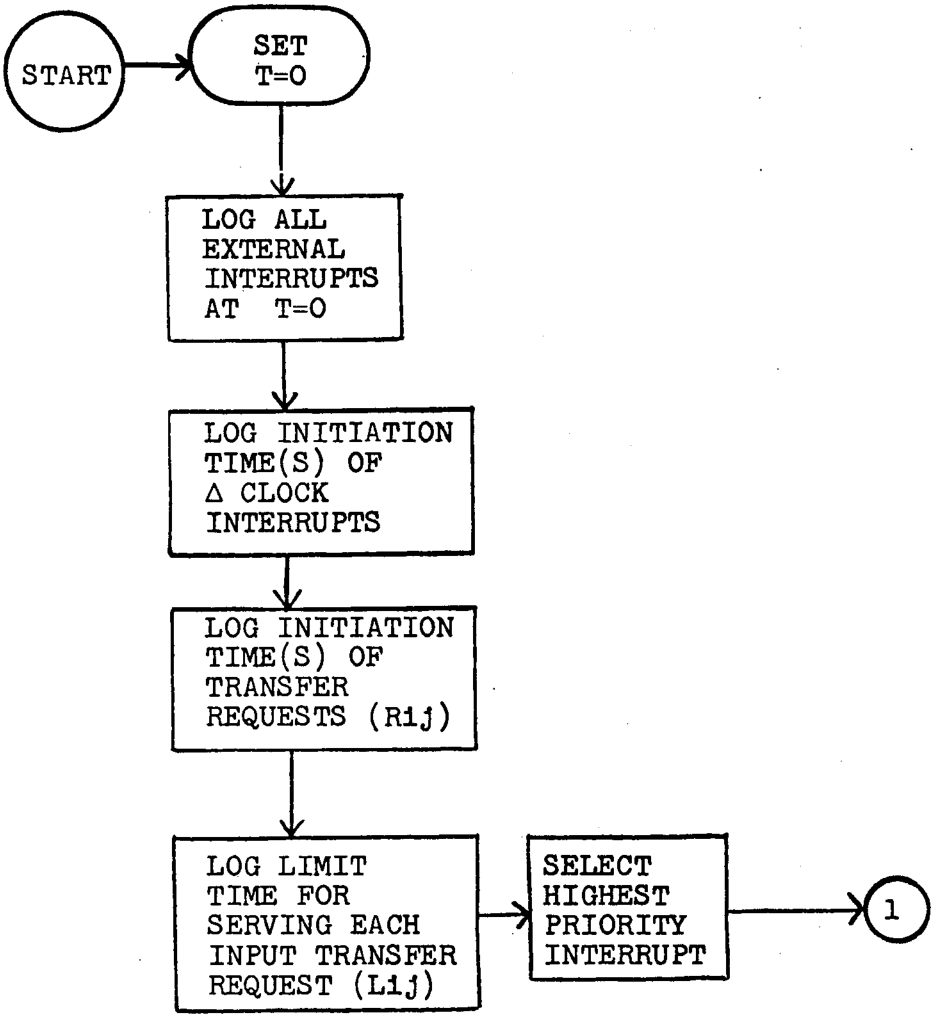
i = Index of communications circuits.

j = Index of Input/Output (I/O) central computer channels.

Example: Rij = Request initiation time to transfer a data character on the i^{th} circuit of I/O channel j.

7 11/16

10 5/8


START
SET T=O
LOG ALL EXTERNAL INTERRUPTS AT T=O
LOG INITIATION TIME(S) OF Δ CLOCK INTERRUPTS
LOG INITIATION TIME(S) OF TRANSFER REQUESTS (Rij)
LOG LIMIT TIME FOR SERVING EACH INPUT TRANSFER REQUEST (Lij)
SELECT HIGHEST PRIORITY INTERRUPT
1

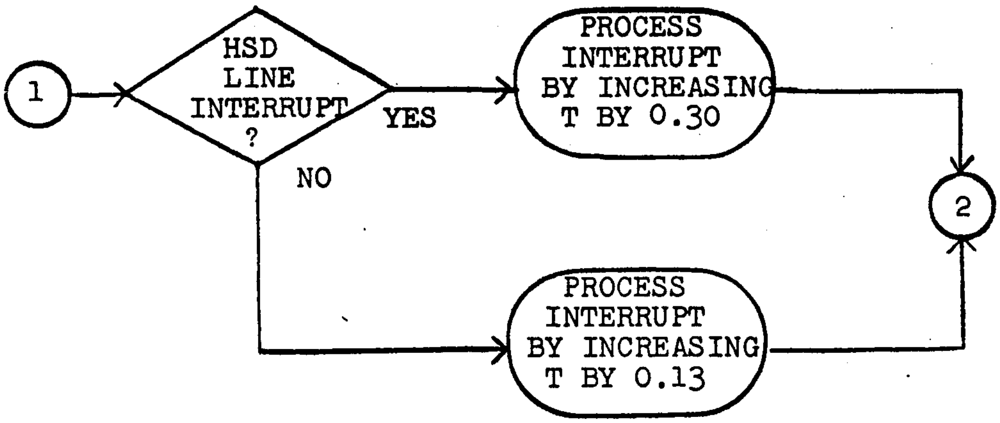
INITIALIZATION

DIAGRAM 1

ISSUE

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET	SHEET

PRINTED IN U.S.A.
E-1812-A-1 (1-61)
OGILVIE PRESS, INC., BROOKLYN 17, N. Y.


1
HSD LINE INTERRUPT ?
YES
NO
PROCESS INTERRUPT BY INCREASING T BY 0.30
PROCESS INTERRUPT BY INCREASING T BY 0.13
2

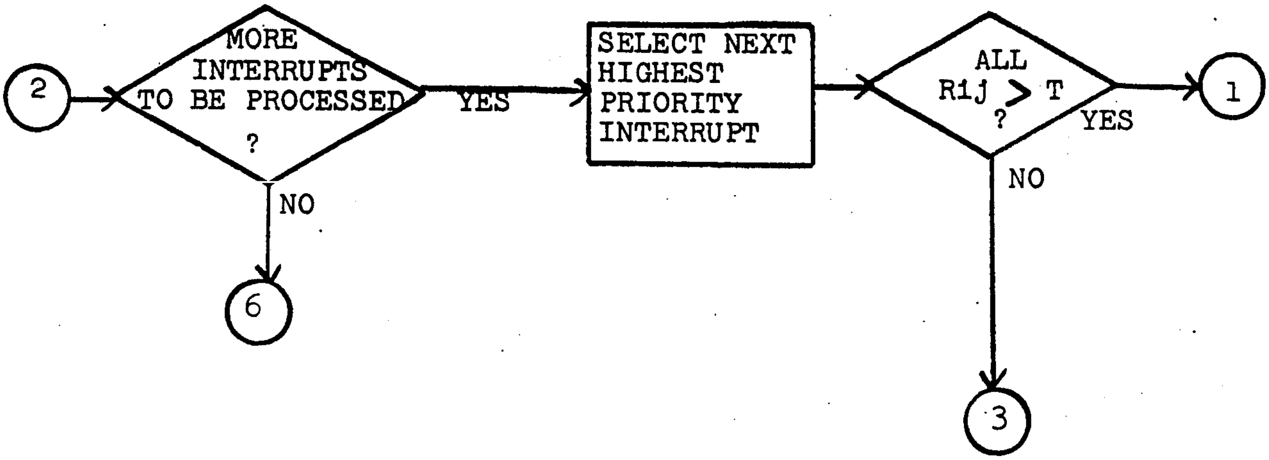
PROCESSING INTERRUPTS

DIAGRAM 2

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET SHEET

7 11/16

10 5/8


2
MORE INTERRUPTS TO BE PROCESSED ?
YES
NO
6
SELECT NEXT HIGHEST PRIORITY INTERRUPT
ALL Rij > T ?
YES
NO
1
3

SELECTING NEXT INTERRUPT

DIAGRAM 3

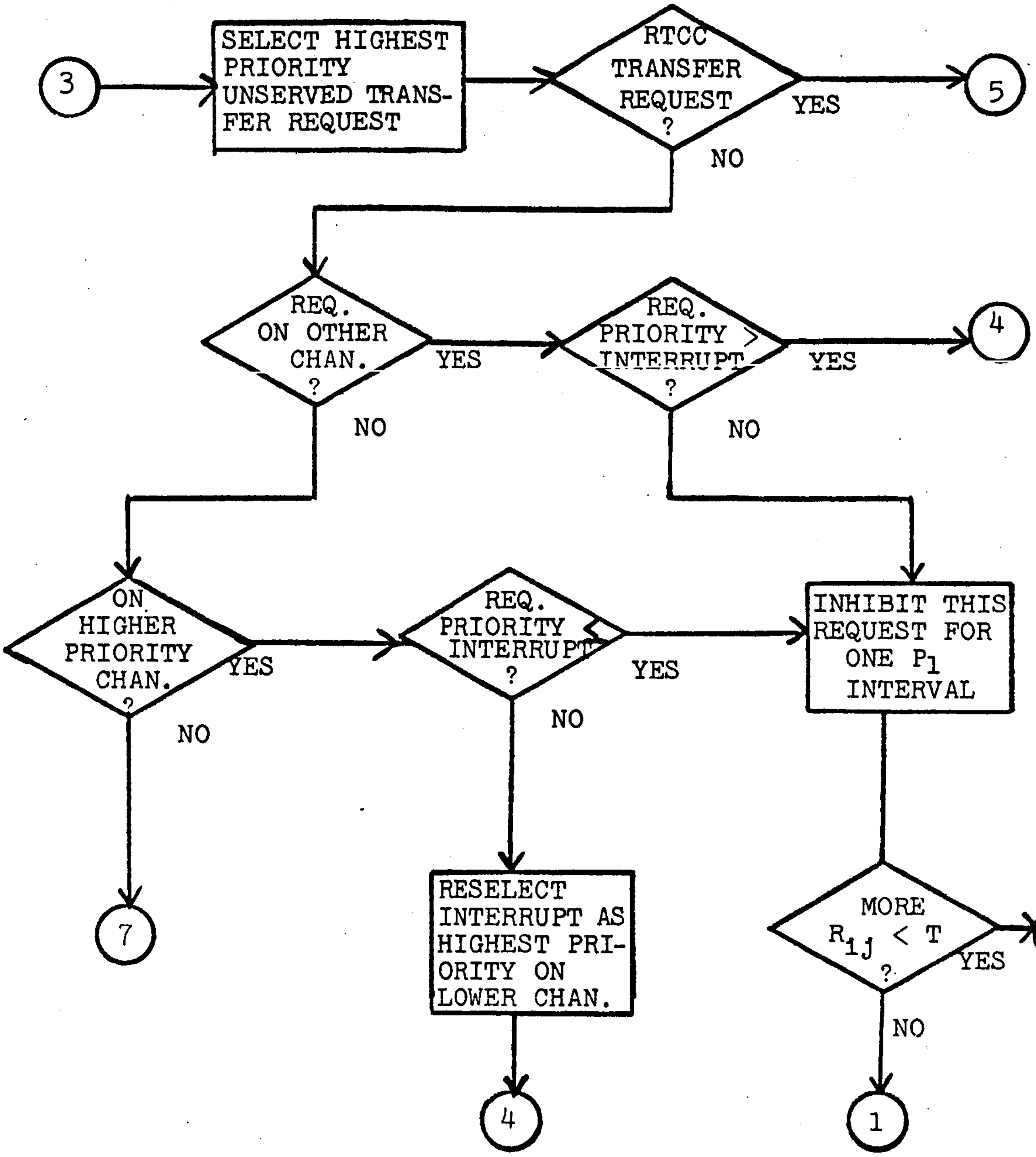
ISSUE

ISSUE	ENGR	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED	
	DRAWN		NO. OF SHEETS PER SET	SHEET

PRINTED IN U.S.A. E-1812-A-1 (1-61)

$7\frac{11}{16}$

$10\frac{5}{8}$


3
SELECT HIGHEST PRIORITY UNSERVED TRANSFER REQUEST
RTCC TRANSFER REQUEST ?
YES
5
NO
REQ. ON OTHER CHAN. ?
YES
REQ. PRIORITY > INTERRUPT ?
YES
4
NO
NO
ON HIGHER PRIORITY CHAN. ?
YES
REQ. PRIORITY < INTERRUPT ?
YES
INHIBIT THIS REQUEST FOR ONE P_1 INTERVAL
NO
NO
7
RESELECT INTERRUPT AS HIGHEST PRIORITY ON LOWER CHAN.
4
MORE $R_{1j} < T$?
YES
3
NO
1

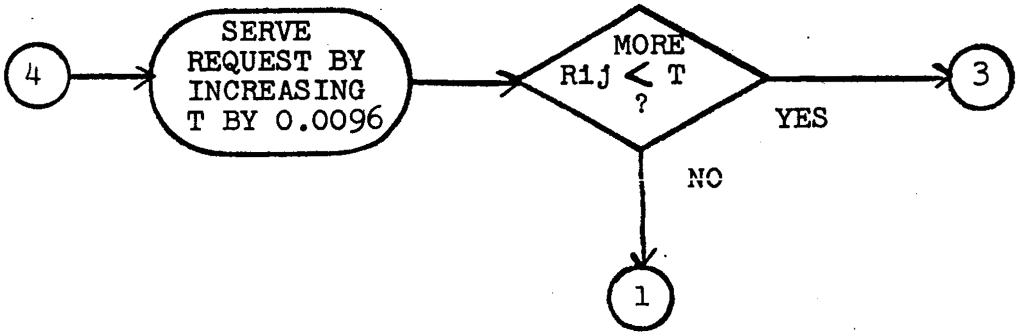
CLASSIFYING TRANSFER REQUEST

DIAGRAM 4

ISSUE

PRINTED IN U.S.A. E-1812-A-1 (1-61)

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET	SHEET



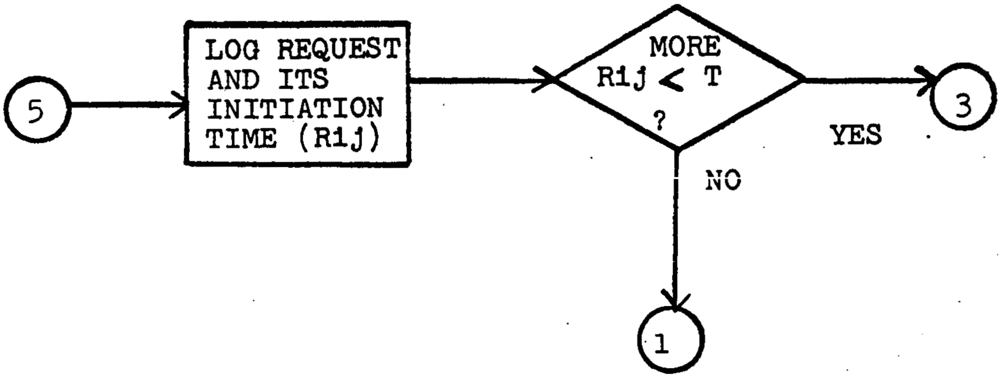
SERVING HSD REQUEST
ON A NON- P_1 BUSY CHANNEL

DIAGRAM 5

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET SHEET

7 $\frac{11}{16}$

10 $\frac{5}{8}$



LOGGING RTCC TRANSFER REQUESTS

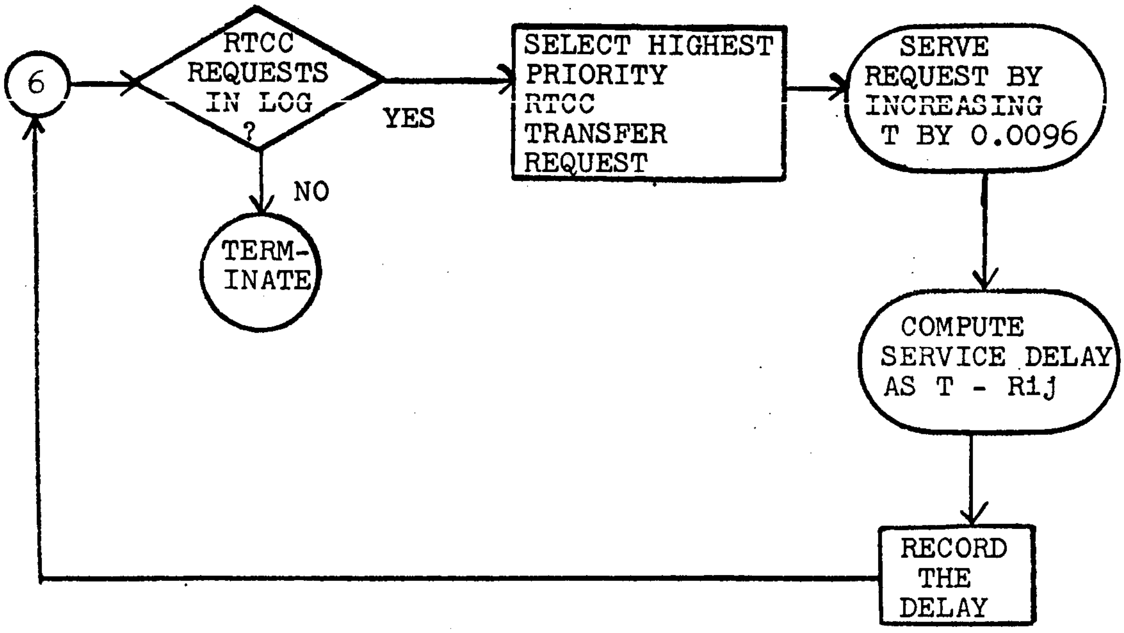
DIAGRAM 6

ISSUE

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET	SHEET

7 11/16

10 5/8


6
RTCC REQUESTS IN LOG ?
YES
NO
TERM-INATE
SELECT HIGHEST PRIORITY RTCC TRANSFER REQUEST
SERVE REQUEST BY INCREASING T BY 0.0096
COMPUTE SERVICE DELAY AS T - Rij
RECORD THE DELAY

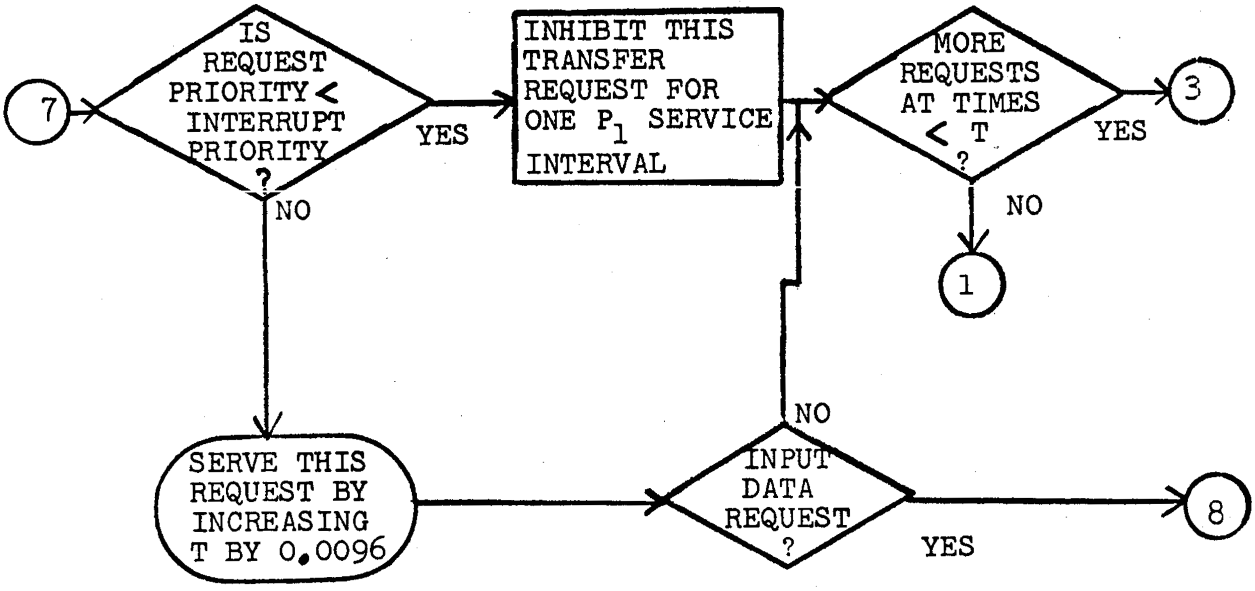
SERVING RTCC TRANSFER REQUESTS

DIAGRAM 7

ISSUE

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET	SHEET

PRINTED IN U.S.A. E-1212-A-1 (1-61)


7
IS REQUEST PRIORITY < INTERRUPT PRIORITY ?
YES
NO
INHIBIT THIS TRANSFER REQUEST FOR ONE P_1 SERVICE INTERVAL
MORE REQUESTS AT TIMES < T ?
YES
3
NO
1
SERVE THIS REQUEST BY INCREASING T BY 0.0096
INPUT DATA REQUEST ?
NO
YES
8

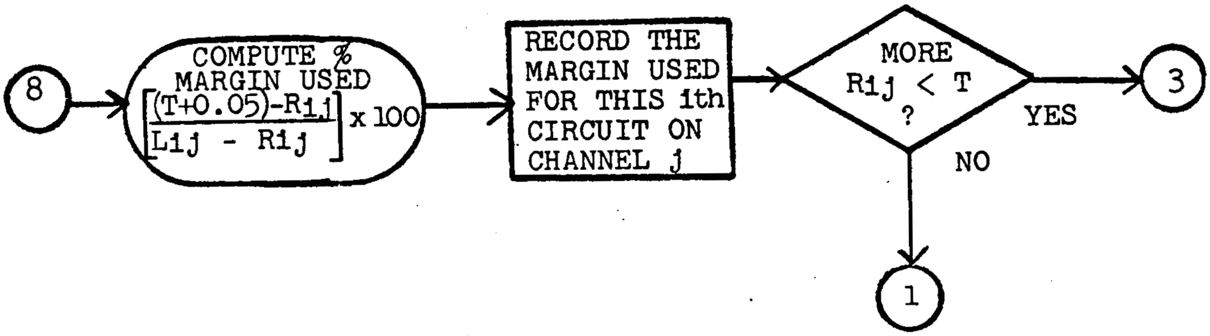
SERVING HSD REQUEST
ON A P_1 BUSY CHANNEL

DIAGRAM 8

ISSUE	ENGR DRAWN	TITLE	BELL TELEPHONE LABORATORIES INCORPORATED NO. OF SHEETS PER SET	SHEET

7 11/16

10 5/8



COMPUTING AMOUNT OF MARGIN USED

DIAGRAM 9

ISSUE

ISSUE

ENGR

DRAWN

TITLE

BELL TELEPHONE LABORATORIES
INCORPORATED

SHEET

NO. OF SHEETS PER SET

PRINTED IN U.S.A.
E-1812-A-1 (1-61)

APPENDIX B

DELAYS TO CHARACTER TRANSFERS WITH THE RTCC

General

In the course of the P_1 Processing Analysis, it was revealed that the most serious consequence of using too much time in P_1 Processing, is the potential loss of data from a high speed data line. In addition, a secondary consequence was noted. This is the potential delay to data from the Real Time Computer Complex (RTCC). This problem arises because of the characteristics of the demand-response interface between the RTCC and the Communications Processor (CP).

1. Demand-Response Interface

The CP performs data transfers with the RTCC in basically the same way that it performs data transfers with the high speed data lines. This operation is described in detail in the initial sections of this memo. There is, however, in the RTCC case, one notable exception. This is the Ready-to-Receive line, which runs between the CP and the RTCC. Only when the CP is ready to accept data transfers from the RTCC, it "activates" the Ready-to-Receive line. When this occurs, data commences to flow in essentially the same way as it does on the high speed data lines.

The Ready-to-Receive line is under the control of a communications worker program, which is entitled the "U94 Handler." In the course of executing the sequence of instructions in this program, the Ready-to-Receive line is activated.

2. Processing Load Effects

If incoming data frame endings are appropriately spaced in time, all of the Processing, associated with each data frame, can be completed before the next frame must be handled. In this case, if it is required, an excursion through the U94 Handler program can be made for each data frame. This will permit activating the Ready-to-Receive line as each frame of data is processed. Therefore, if there are any character transfers from the RTCC, they can be accommodated as soon as the Ready-to-Receive line is activated.

If on the other hand, the data frames are not spaced in time, but instead have coincident ending times, a different situation results. In this case, the frame endings will, of course, cause a coincident group of external interrupts to occur. Therefore, it will be necessary to perform a "Record Interrupt Sequence" (P_1) of instructions for the entire group of frames. This must be done before any other programs can be called upon. This leads to the fact that all P_1 Processing for this coincident group must be carried out, before we can get to the U94 Handler program.

Consequently, any data characters destined for the CP, from the RTCC, must wait in the RTCC until P_1 Processing is completed. When P_1 is complete, the U94 Handler program can be brought into play; the Ready-to-Receive line can be activated, and data transfers from the RTCC to the CP can occur.

3. The Combined Gemini Live and Simulation Loads

This section is dedicated to an examination of the effect of the foregoing characteristics on the combined P_1 Processing load. The combined Gemini live plus simulation load used in the body of this memorandum, was based upon coincident frame endings. Therefore, if succeeding frames of data were available for transfer from the RTCC after T zero, they would have to be delayed. The amount of delay, would be approximately the amount of time required to complete the P_1 Processing of the load. As shown earlier, P_1 Processing of the combined load required about 4.4 ms of time. Therefore, if transfer requests from RTCC, occurred at T zero + δ, where δ is infinitesimally small, the transfers would be delayed about 4.4 ms.

The flow of data characters in the other direction, i.e., CP to RTCC has not been quantitatively considered in this analysis. The main reason for this is that the Ready-to-Receive line in this direction is under the control of the

RTCC. Therefore, in order to perform a quantitative analysis of CP to RTCC transfers, the analysis would have to include the RTCC load, in relation to its activation of the Ready-to-Receive line. However, this is beyond the scope of this work. Qualitatively, any delays to character transfers in this CP to RTCC direction, are typically expected to be considerably less than those in the other direction.